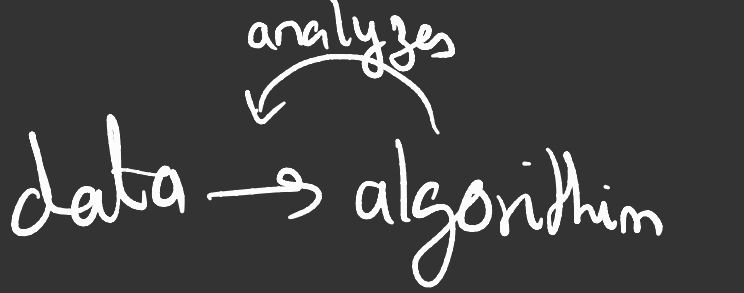
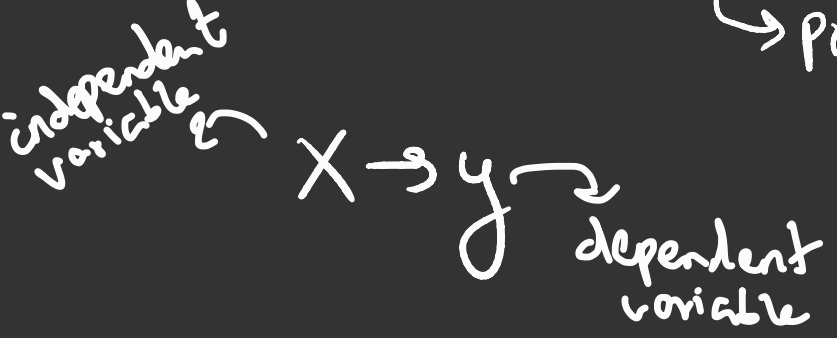
Machine Learning
free Code Camp

What is Machine learning?

Ans- Computer programs that uses algorithms to analyze data and make intelligent predictions based on the data without being explicitly programmed

Data is given to the algorithm which analyzes it and gives out predictions





X is the input feature

y is the output variable

$f(x) \longrightarrow y$

$f(x_1, x_2, x_3) \longrightarrow y$

eg- Email spam detection system

T → 0/1 (0 is not spam, 1 is spam)

E → Experience

P → Performance

* A computer program is said to learn from experience E with respect to some task T and some performance measure P.

improves with experience E

How it works?

Ans) ★ Study the problem
★ Train the algorithm.
★ Evaluate it
★ if it is good
★ Else, Analyze the error

Eg: Spam detection

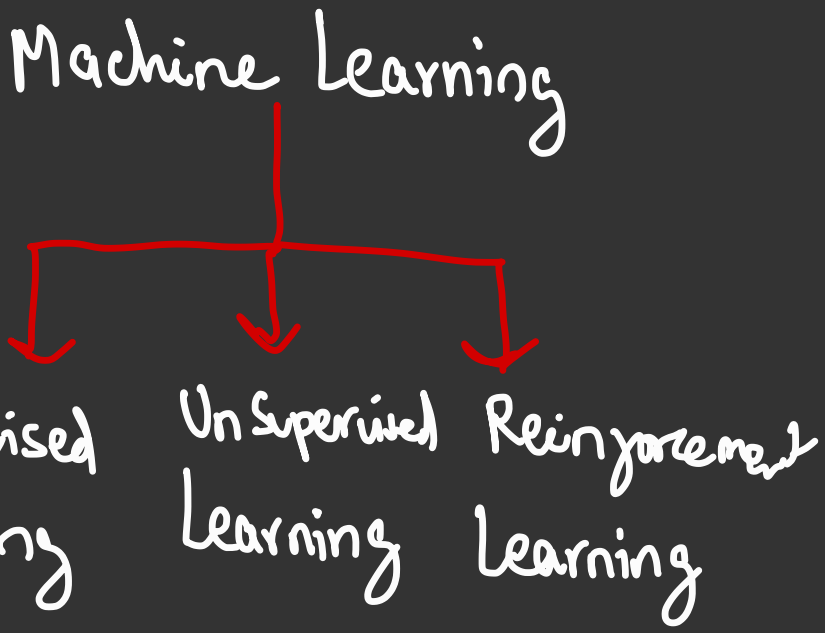
→ Study the problem
→ f(x) → algorithm
→ evaluate

★ ML is a very iterative process

→ Launch the system

→ Error Analysis

Types of Machine Learning Systems



Supervised Learning :-

In this the data we feed is properly organised with proper Labels such as -> CSV, excel files etc.

In which the data are labeled and we know what our output should be having the relationship b/w input values "x" and output values "y"

eg- House price prediction

(Size of the house) → (Price of the house)

Continous

X — input feature

Y — target variable

house → house

in supervised Learning we're given

$\{(x_i, y_i)\}_{i=1}^{m}$ → m → no.where m is the data.

Unsupervised Learning :-

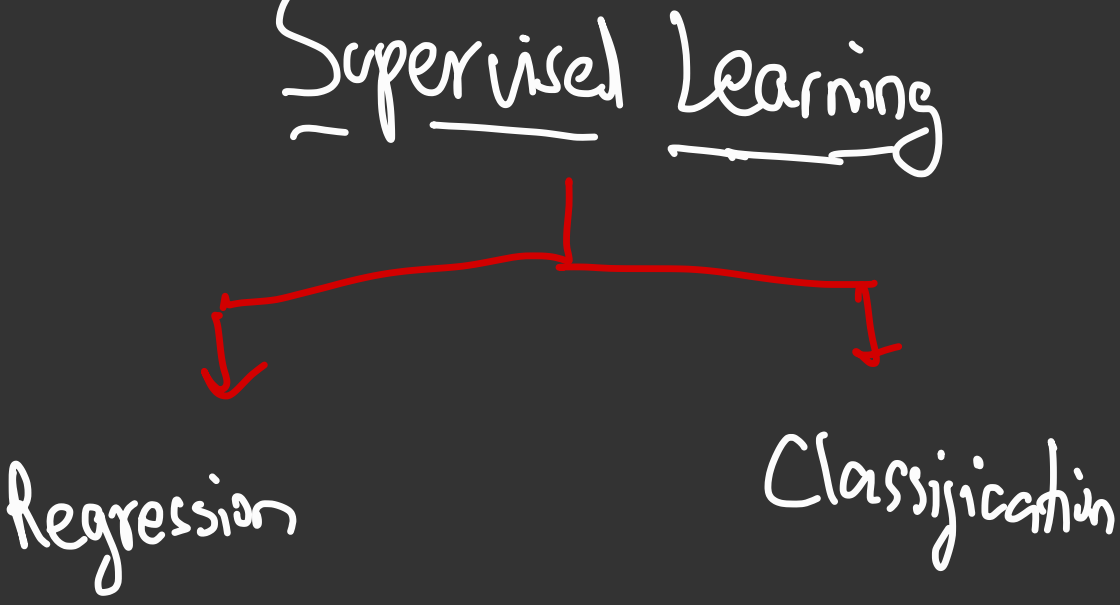
In this the data we feed is not organised nor properly labelled, And we don't know what our output should like and there is not any kind of relationship between input variables and output variables, we have to recognize patterns based on the data.

Eg- working on data like

images, pdfs, audio etc.

$\{(x_1, x_2, x_3, x_4 \ldots x_m)\}$

Types of Supervised Learning problems :



Regression - In this type of supervised learning we're supposed to predict a specific number from a continous data.

eg:- Predicting House prices

Classification:- In this type of Supervised learning we're supposed to predict categories of data from categorical data

eg -> Predict is it a apple/orange, cat/dog, yes/no problems

Dividing our data :-

we divide our data into 2/3 sets ⇒ Train_test_validation
Train_test

Train Set - The sample of data used to fit the model and train it

Validation Set :- The sample of data used to provide an unbiased evaluation of a model fit on the training set

while tuning model hyperparameter. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration.

Test Dataset: The sample of data used to provide an unbiased evaluation of a final model fit on the training data set

* Sometimes people use test-train-validation set to be extra cautious

* Other times people use only train-test sets using test as both validation and test.

* There is no optimal split percentage but the common split percentage include:-

→ Train: 80% , Test: 20%.

→ Train: 70% , Test: 30%.

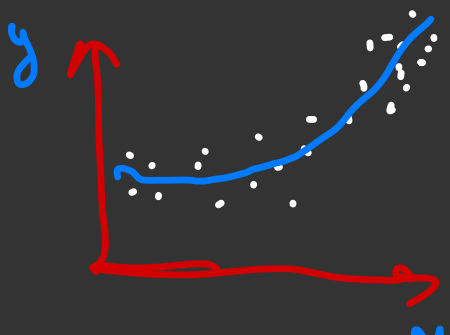
→ Train: 80%, Test: 10%, validation: 10%.

→ Train: 70%, Test: 15%, validation: 15%.

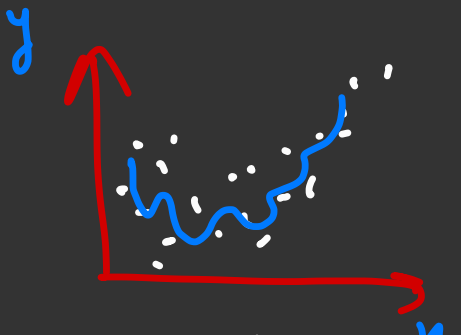
Overfitting the training data:

It means the model performs well on the training data, but it does not generalize well.

Solution to this problem is to gather more data, to reduce noise in the training data, to use regularisation.



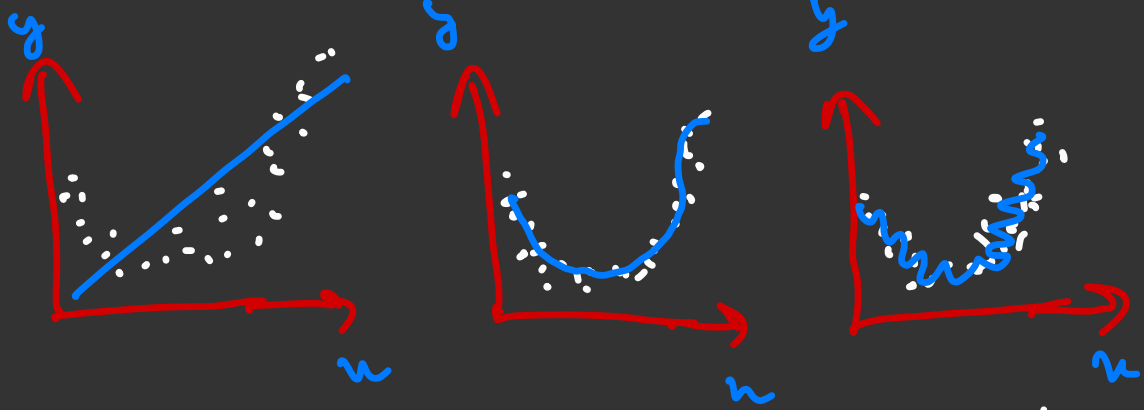
Just right



overfitting

Underfitting the training Data:-

It means the model performs bad on the training data, so it's obvious it will also perform baldly on testing data.

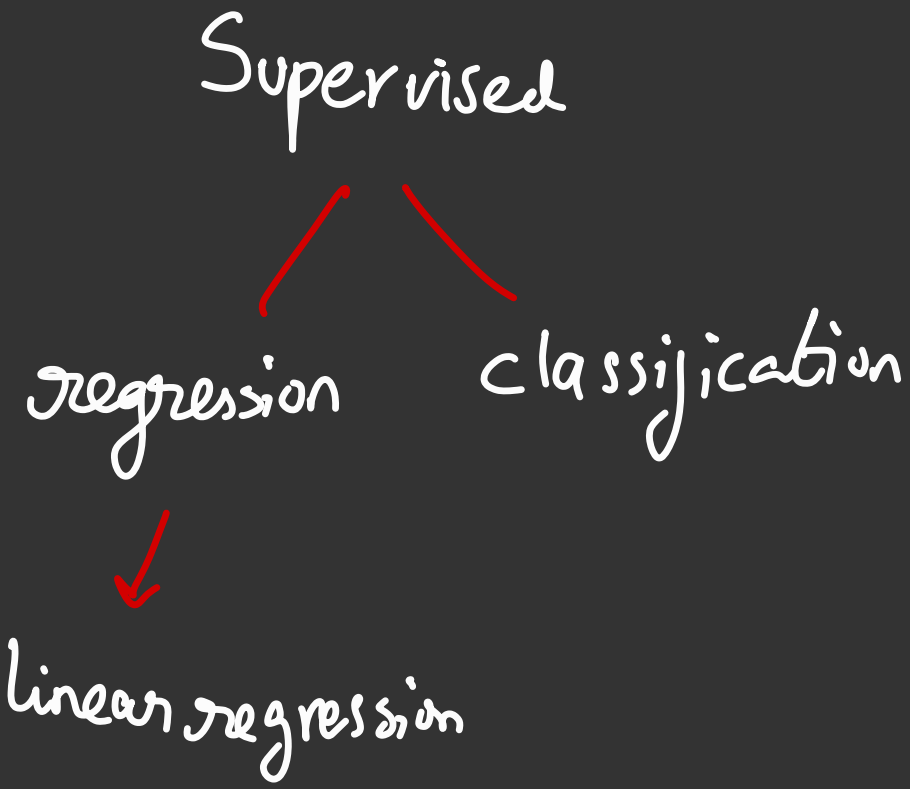


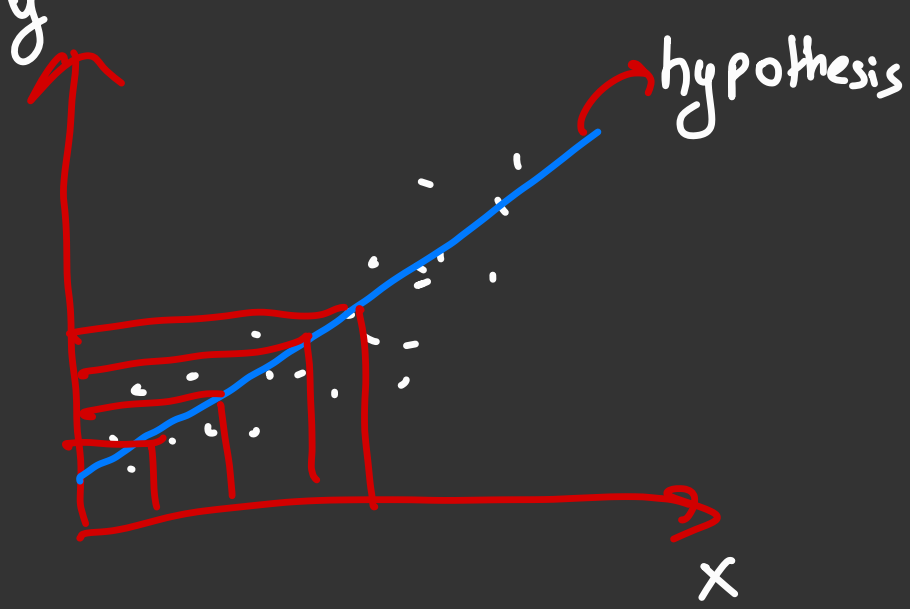
underfitted

goodfitt
Robust

Overfitted

Linear Regression :-





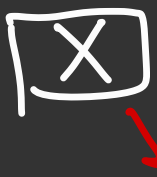
hypothesis is a straight line as near to all the data points

hypothesis function :-

$$y = \theta_0 \cdot x_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2 \cdots + \theta_n \cdot x_n$$

* x_0 is always 1

the y intercept



Every X is a different input feature

$$\therefore f(n) = \theta_0 \cdot x_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2 \ldots + \theta_n \cdot x_n$$

$$= y$$

Eg → We have to predict the price of house from size of the house and no. of jars in it.

let $\theta_0 = 2$

$\theta_1 = 3$

$\theta_2 = 4$

$\therefore \Rightarrow$ 2 + 3 · size of house + 4 · no. of fans

$= \boxed{y}$

We have the input features from the data, we only need to figure out θ for each input feature.

$$f(x) = \theta_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2 + \cdots \theta_n \cdot x_n$$

Vector form:

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$

$$f(x) = \theta^T \cdot X$$

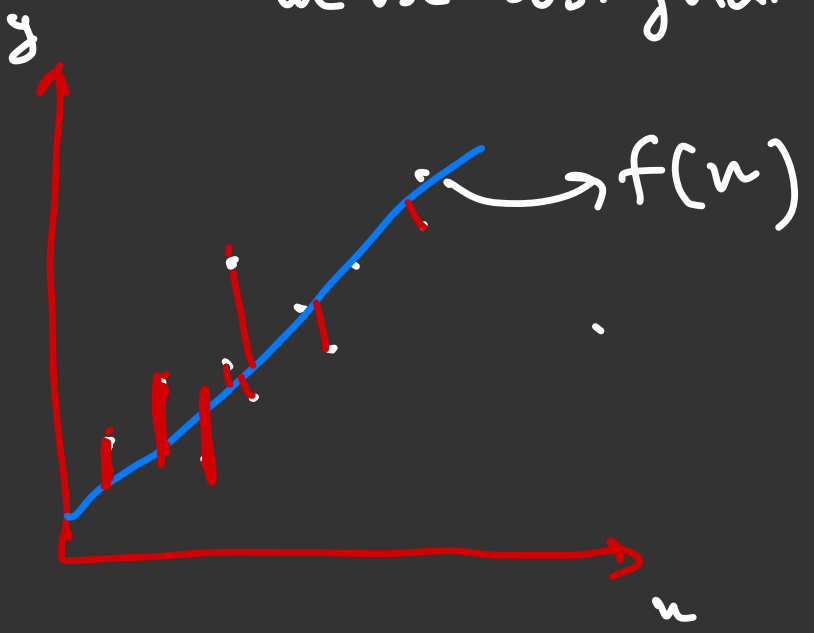
(np.dot(theta, x))

or

$$f(x) = \sum_{i=0} \theta_i^T \cdot X_i$$

To evaluate your model :

we use cost function



We find the distance b/w predicted and actual values.

i.e ⇒ Predicted value - actual value

$$J(\theta) = \frac{1}{m} + \sum_{i=1}^{m} (y - \hat{y})^2$$

MSE (mean squared Error)

m → No. of data points

$\hat{y}$ → $f(x) = \sum_{i=0} \theta_i x_i$ → Predicted value

y → actual values

 if square root $J(\theta)$ i.e $\sqrt{J(\theta)}$ it is called RMSE.

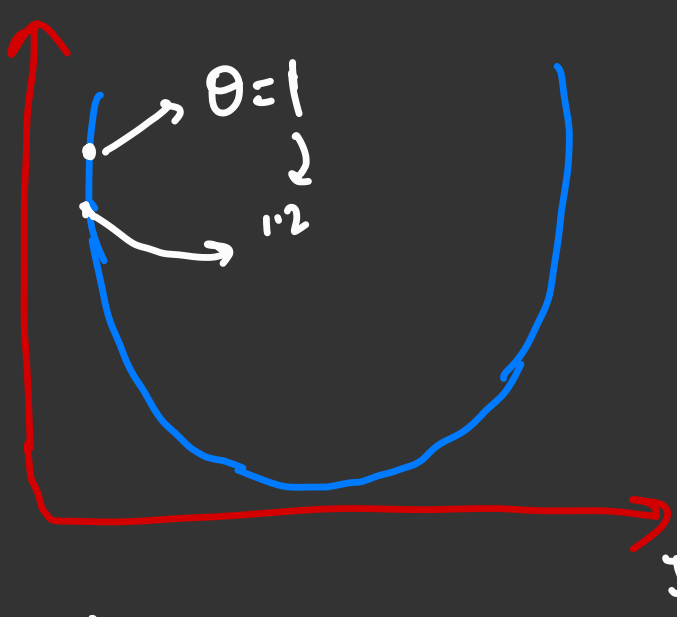
(root mean squared error).

higher RMSE or MSE, the badder is the model

How to find the most optimal θ?

We use gradient descent algorithm which is also called optimization algorithm)

θ.



graphing gradient descent algorithm

it keeps tweaking the θ values until the cost function ≈ 0

$J(\theta) \approx 0$

to do that we find out the

$\frac{\partial}{\partial} J(\theta)$ => partial derivative of $J(\theta)$

$$\frac{\partial}{\partial} J(\theta) = \frac{2}{m} + \sum_{i=1}^{m} (\hat{y} - y)^2$$

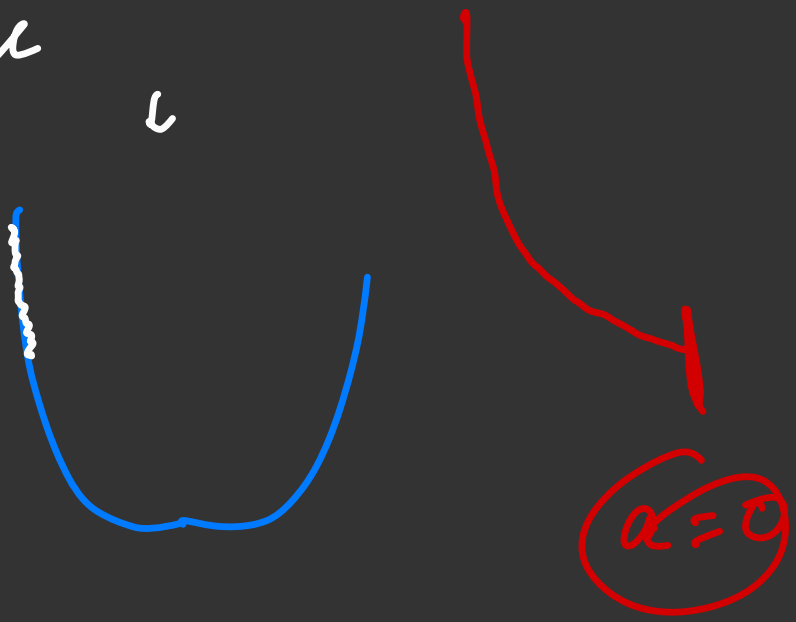
$$\theta := \theta - \alpha \frac{\partial}{\partial} (J(\theta))$$

Algorithm for gradient descent

new θ | old θ | Learning rate | old θ

* if α is too large it will never converge

* if θ is to small : it will never diverge



* Optimal α's

↓

0.1 for large dataset
0.01 for medium to smaller
0.001 for smaller
0.0001 for very small dataset

You can tune it using RandomCV

Vector form-

$$\frac{\partial}{\partial} J(\theta) = \begin{bmatrix} \frac{\partial}{\partial} J(\theta_0) \\ \frac{\partial}{\partial} J(\theta_1) \\ \vdots \\ \frac{\partial}{\partial} (J\theta_n) \end{bmatrix}$$

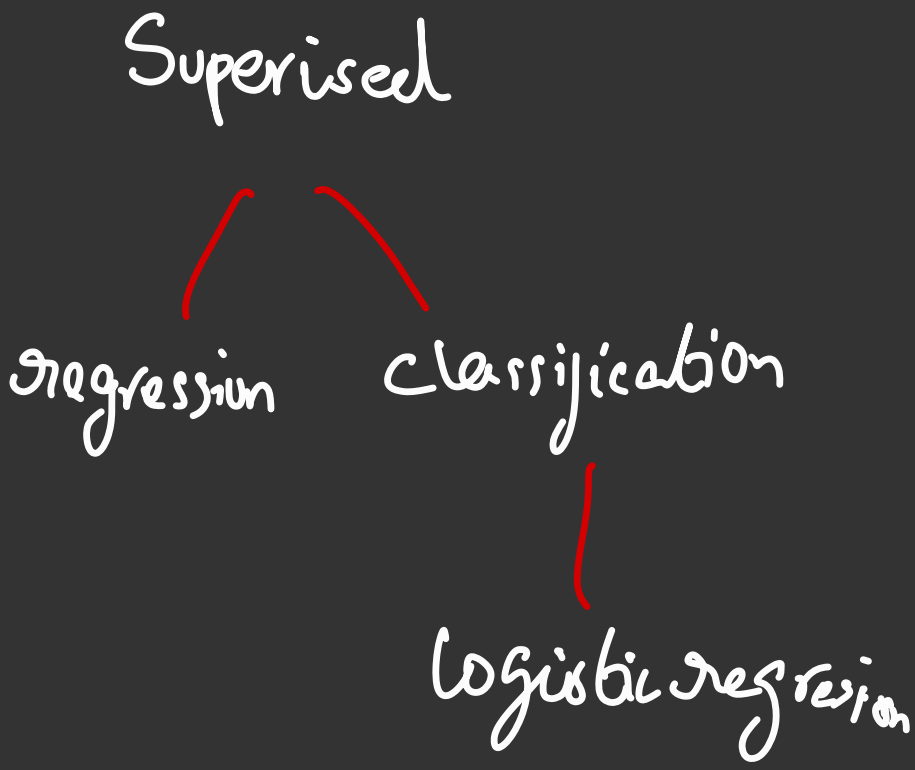
Normal Equation

$$\theta = (X^T \cdot X)^{-1} X^T y$$

Polynomial Regression :-

in this we transform our data to be fitting over linearly

Logistic Regression



$$h(x) = \theta_0 \cdot x_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2 \cdots + \theta_n x_n$$

↓

$$\hat{y} = \sigma(h(x))$$ Sigmoid function

↓

restricts the value b/w 0 and 1

$$\therefore \sigma(h(x)) = \frac{1}{1 + e^{-h(x)}}$$

Cost function :-

$$J(\theta) = -1 \left(y^{(i)} \log(h(x^{(i)})) + (1-y^{(i)}) \times \log(1-h(x^{(i)})) \right)$$

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} y^{(i)} \log(h(x^{(i)})) + (1-y^{(i)}) \log(1-h(x^{(i)}))$$

$$h(x) = g(h(x))$$

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(h^{(i)} - y^{(i)}\right) \cdot x_j$$

$$\theta_j = \theta_i - \alpha \frac{\partial}{\partial} J(\theta)$$

Linear regression from scratch:

Approximation:

$$\hat{y} = wx + b$$

$w \Rightarrow$ slope

$b \Rightarrow$ bias/intercept

$\hat{y} \Rightarrow$ predicted values

$x =$ data

Cost function:

$$MSE = J(w,b) = \frac{1}{N}\sum_{i=1}^{N}(y_i - (wx_i + b))^2$$

MSE => Mean Squared Error

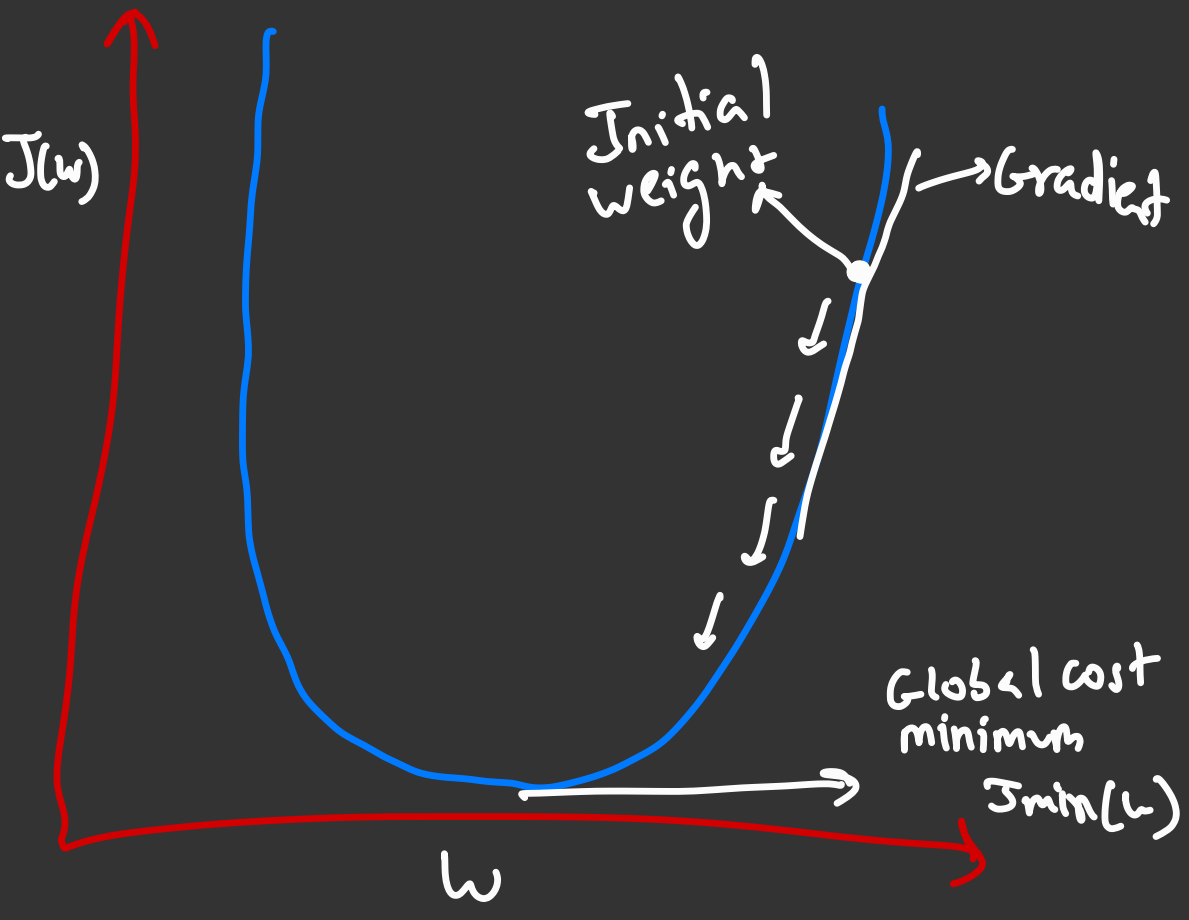
N => Total length of data

y_i => actual values

We use gradient descent to find the w and b values which gives the lowest MSE

$$J'(w,b) = \begin{bmatrix} \frac{df}{dw} \\ \\ \frac{df}{db} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{N} \sum -2x_i (y_i - (wx_i + b)) \\ \frac{1}{N} \sum -2 (y_i - (wx_i + b)) \end{bmatrix}$$



(same graph b/w J(b) and b

Updated values become

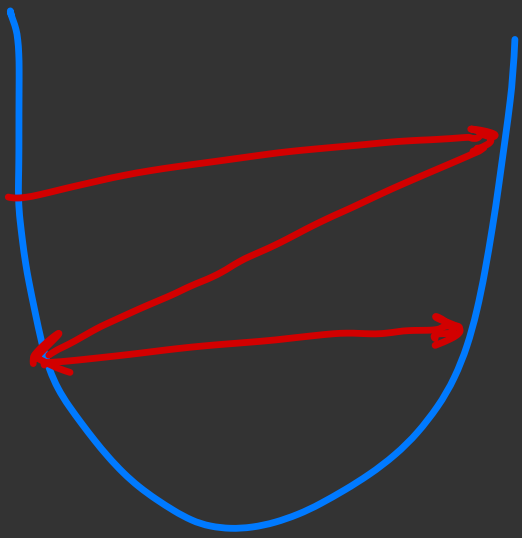
$$w = w - \alpha \cdot dw$$

$$b = b - \alpha \cdot db$$

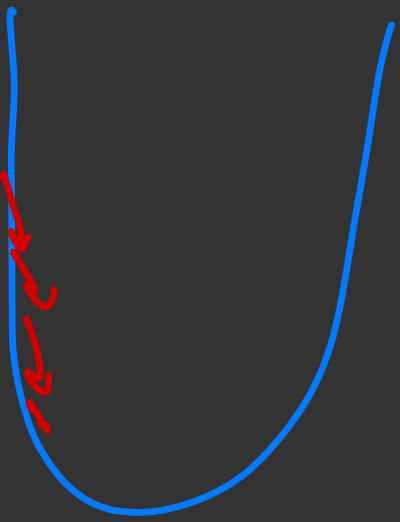
$\alpha \Rightarrow$ Learning rate

$$\frac{dJ}{dw} = dw = \frac{1}{N} \sum_{i=1}^{N} -2x_i \left(y_i - (wx_i + b)\right)$$

$$\frac{dJ}{db} = db = \frac{1}{N} \sum_{i=1}^{N} -2\left(y_i - (wx_i + b)\right)$$



big learning rate



small learning rate